MrRoboto

1 How to setup system

If you're using the robot lab's computer skip to **??**.

1.1 Initial computer settings

1. Install ubuntu 12.04 on your computer.
2. Install ROS GROOVY according to: `http://wiki.ros.org/groovy/Installation/Ubuntu`
3. Install git, type:
   $ sudo apt-get install git (in a terminal window)
   Password is *ubuntu* for the computer in the robot lab.
4. Clone the repository of the project. cd* into a folder where you want the folder containing the code. Type:
   $ git clone git@github.com:matni796/robot-security-tsbb11
   (*cd is a terminal command. If your not familiar with navigation in terminal, see `http://linuxcommand.org/lc3_lts0020.php`)
5. Source your own created packages in .bashrc: type: gedit /.bashrc add the line: source *wherever_path_you_put_repository/robotsecuritytsbb11/catkin_ws/src/devel/setup.bash*
6. Install ROS, type:
   $ sudo apt-get install ros-groovy-desktop-full
7. Install freenect, type:
   $ sudo apt-get install ros-groovy-freenect-stack
   $ sudo apt-get install ros-groovy-freenect-launch
8. Install arm navigation (used for visualization)
   $ sudo apt-get install ros-groovy-arm-navigation-experimental
9. Disable gspca kernel according to: `http://openkinect.org/wiki/Getting_Started`
10. cd into the installed folder and into catkin_ws and type catkin_make to build.

1.2 Setup of DX100 controller

1. Make sure the controller's version supports MotoPlus applications, should be a version ending with -14. Current version (20131206) is DS3.53.01A-14.
2. Load parameters. The file ALL.PRM can be found in git repo, under the folder robot/. Transfer it to a cf card or usb. Start the controller regularly. Go into management mode (see below). Then go to:

   EX MEMORY → FOLDER, set folder where you put ALL.PRM.

EX MEMORY → LOAD → PARAMETERS → BATCH PARAMETERS ALL.PRM

You might need to do a safety reset of the flash device. This is done by starting up the controller in Maintenance mode. In Maintenance Mode, enter Management mode. INITIALIZE → system flash safety reset, might take a while, wait for beep. Shut off controller and restart it regularly.

NOTE: Might change the setup of the controller. Should be done with caution. Contact Yaskawa if uncertain.

3. To install MotoPlus application, follow the tutorial on `http://wiki.ros.org/motoman_driver/Tutorials/InstallServer`.

   - For step 3 in tutorial, see PDF file *MotoPlus Application Installation* in folder *robot/*. Follow the instructions on step 2.1. The application file: *MpRosSia20.out* to be loaded is also in the in the *robot/* folder.

     NOTE: We have used an .out file hardcoded for a SIA20D robot. This version does not come with the motoman files from ROS.

   - For step 4 in tutorial, transfer *INIT_ROS.JBI* to CF card or USB device. File can be found under *../catkin_ws/src/motoman/motoman_driver/Inform/DX100/*. Start the controller regularly. In menu, go to:

     Ex MEMORY → FOLDER move to the folder where you put *INIT_ROS.JBI* file. Then:

     Ex Memory → LOAD → job.

1.3 Setup of computer network settings

1. Connect the computer to the controller via ethernet cable. Use the output CN104 on the YCP01 board.
2. Edit settings for wired/local network on computer. Set computer wired network address to 192.168.255.9 and netmask 255.255.255.0.

   NOTE: On the computer, running Ubuntu 12.04, in the robot lab, there is a profile for the connection named "DX100". When connecting the controller to the computer via ethernet, choose this.

2 Run program

When the setup is done follow these instructions to run the system.

2.1 Establish connection between controller and computer

The complete instructions for establishing a connection with the controller can be found at: `http://wiki.ros.org/motoman_driver/Tutorials/Usage`. Run the following in a separate terminal. Hint: *Ctrl + Shift + T* opens a new tab.

- *$ roscore*
- *$ rosparam load ...catkin_ws/src/motoman/motoman_config/cfg/sia20D_mesh.xml robot_description*
- *$ roslaunch motoman_driver robot_interface_streaming_dx100.launch robot_ip:=192.168.255.1*

NOTE: Currently it is not possible to send motion commands to the robot. This is due to developers lack of implementation skills.

2.2 Launch system

While still running *roscore* and the robot node. Run the following comands, each in a seperate terminal. Use *Ctrl+Shift+T* to add a tab to existing terminal window.

- *$ roslaunch freenect_launch freenect.launch*
- *$ rosrun background_modelling background_modelling*
- *$ rosrun clustering clustering*
- *$ rosrun calibration calibration*
- *$ rosrun distance_calc distance_calc*
- In order to set the static transform between the calibration pattern and the robot base. In terminal, type:
  $ rosrun tf static_transform_publisher x y z yaw pitch roll base_link pattern 100

  where x, y, z and yaw, pitch, roll has to be specified. With the setup in our project they were: $[0, 0, 0, \pi, \pi, \pi]$. This might need to be tweaked to get a good calibration.
- To obtain visualization, type:
  $ roslaunch sia20d_mesh_arm_navigation planning_scene_warehouse_viewer_sia20d_mesh.launch
  This will launch the visualisation environment RViz.

NOTE: In each terminal there will be data printed about that function's current state.

2.3 Visualization

RViz provides a wide range of possibilities for visualization. To begin with there are some fundamental settings that always needs to be performed.

- Under Global Options, choose /base_link as Fixed Frame.
- Add a PointCloud2 and choose /objects as Topic.
- For one of the Markers choose the Topic /closest_line
- Add a camera
  - Choose an image topic, preferably mono

- Set "overlay alpha" to 1
- Set queue size to 1
- Choose an appropriate image rendering.
  Note: The camera matrix used for visualization is the default. The backprojection will not coincide perfectly with the image